Robust Localization Against Outliers in Wireless Sensor Networks

QINGJUN XIAO, KAI BU, ZHIJUN WANG, and BIN XIAO, The Hong Kong Polytechnic University

In wireless sensor networks, a critical system service is the localization service that determines the locations of geographically distributed sensor nodes. The raw data used by this service are the distance measurements between neighboring nodes and the position knowledge of anchor nodes. However, these raw data may contain outliers that strongly deviate from their true values, which include both the outlier distances and the outlier anchors. These outliers can severely degrade the accuracy of the localization service. Therefore, we need a robust localization algorithm that can reject these outliers. Previous studies in this field mainly focus on enhancing multilateration with outlier rejection ability, since multilateration is a primitive operation used by localization service. But patch merging, a powerful operation for increasing the percentage of localizable nodes in sparse networks, is almost neglected. We thus propose a robust patch merging operation that can reject outliers for both multilateration and patch merging. Based on this operation, we further propose a robust network localization algorithm called *RobustLoc*. This algorithm makes two major contributions. (1) RobustLoc can achieve a high percentage of localizable nodes in both dense and sparse networks. In contrast, previous methods based on robust multilateration almost always fail in sparse networks with average degrees between 5 and 7. Our experiments show that RobustLoc can localize about 90% of nodes in a sparse network with 5.5 degrees. (2) As far as we know, RobustLoc is the first to uncover the differences between outlier distances and outlier anchors. Our simulations show that RobustLoc can reject colluding outlier anchors reliably in both convex and concave networks.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms: Design, Algorithms, Reliability, Security

Additional Key Words and Phrases: Wireless sensor networks, sensor node localization, outlier distance rejection, outlier anchor rejection

ACM Reference Format:
Xiao, Q., Bu, K., Wang, Z., and Xiao, B. 2013. Robust localization against outliers in wireless sensor networks. ACM Trans. Sensor Netw. 9, 2, Article 24 (March 2013), 26 pages.
DOI: http://dx.doi.org/10.1145/2422966.2422981

1. INTRODUCTION

In wireless sensor networks, location information is critical for a wide range of applications and protocols. For example, environmental monitoring applications need the node locations to make sensing data geographically meaningful. Routing protocols need the location knowledge to enable geographical routing and reduce communication cost. But it is not an easy task to furnish each node with its location in a large-scale network. Manually configuring each node is too troublesome, and equipping each node with a GPS receiver is expensive. Therefore, it becomes a research issue to find cost-effective methods to derive the location knowledge in a sensor network, which is called the *network localization problem* [Whitehouse et al. 2005; Eren et al. 2004; Xiao et al. 2008;

This work was partially supported by the project HK RGC PolyU 5314/10E.
Author's addresses: Q. Xiao, K. Bu, Z. Wang, and B. Xiao, Department of Computing, The Hong Kong Polytechnic University; email: {csqjxiao, cskbu, cszjwang, csbxiao}@comp.polyu.edu.hk.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2013 ACM 1550-4859/2013/03-ART24 $15.00
DOI: http://dx.doi.org/10.1145/2422966.2422981

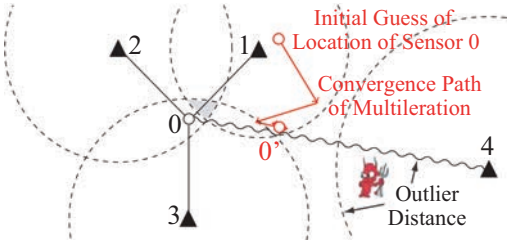


Fig. 1. Impact of outlier link [4, 0] on localization accuracy of sensor 0.

Savvides et al. 2003; Priyantha et al. 2003; Goldenberg et al. 2006; Horn et al. 1988; Wang et al. 2008; Shang and Ruml 2004; Moore et al. 2004; Niculescu and Nath 2003; Li and Liu 2007; Lim and C. 2005].

In this research field, the mainstream studies assume that the distances between neighboring nodes are accurately measured and then derive node locations accordingly. Typical distance-measuring techniques or ranging techniques include ultrasonic time-of-arrival (TOA) and ultra-wideband radio TOA. However, among these distance measurements, there inevitably exist erroneous measurements whose errors are abnormally larger than ranging noise. We call them *outlier distances* or outlier links. Outlier distances can have three anomalous causes: (1) internal malfunctions of ranging hardware, (2) malicious attacks, for example, node compromise, or (3) environmental interferences to the ranging signals. For example, due to non-line-of-sight propagation of sound signals, the ranging devices based on ultrasonic TOA may generate outlier distances with strongly enlarged estimates [Whitehouse et al. 2005].

Outlier distances can severely degrade the accuracy of network localization algorithms. For example, in Figure 1, the outlier link [4, 0] can bias the location estimate of node 0 to be at erroneous position 0′. Specifically speaking, in Figure 1, nodes 1, 2, 3, 4 are anchor nodes whose locations are known a priori. The sensor 0 needs to localize, and it can measure distances to these anchors. Each distance measurement is drawn as a dashed circle that is centered at a corresponding anchor node. Among the four links, link [4, 0] is an outlier whose measurement is abnormally smaller than its true value. Misled by this outlier link, the location estimate of sensor 0, when applying the well-known multilateration algorithm, will converge to the severely inaccurate position 0′, following the red polyline.

We thus need to reject the harmful outlier distances during network localization in order to achieve *robust network localization*. The previous studies in this field are focused on adding outlier rejection ability to multilateration, which is called *robust multilateration* [Liu et al. 2005; Kung et al. 2009; Li et al. 2005; Kiyavash and Koushanfar 2007; Wang et al. 2007], because multilateration is a primitive operation that can be applied iteratively to localize a network. Here, iteratively means that a node, after it has been localized, can be upgraded to an anchor which in turn can help localize other nodes [Savvides et al. 2003]. If any node can measure distances to at least three anchors, then this node can derive it own location by multilateration.

However, the previous research based on iterative multilateration becomes weak and even powerless for localization in sparse networks where the node degree can be six or less. In sparse networks, we must resort to another primitive operation called *patch merging*, where a patch is a group of nodes forming a rigid structure [Eren et al. 2004]. This operation can merge two small patches to generate a larger patch if the relative motion of these two patches can be constrained by enough connectivity between them [Horn et al. 1988; Wang et al. 2008; Shang and Ruml 2004; Moore et al. 2004]. In

sparse networks (with node degrees between 5 and 7), this operation can improve the percentage of localizable nodes on average by 30%.

However, the patch merging operation is still vulnerable to outlier distances due to its nature of least square estimator. There is no previous work to invent *robust patch merging* that can reject outliers during patch merging. Thus, we propose such a robust operation as the basis of robust network localization. This robust patch merging provides two advantages over the traditional robust multilateration: (1) it is more powerful, because it can reject outlier links even in sparse networks; (2) it is technically more generalized, since it can reject outliers either when merging two patches or during multilateration (i.e., a special case of patch merging which merges a single node with a patch). Our basic idea of implementing this robust patch merging is to find a *consistent* subset among the links connecting the two patches. This subset contains no outlier links that can be detected (i.e., the residual of each link in this subset is below a predefined threshold).

Even with robust patch merging, there still exists a challenge towards robust network localization. There is a precondition for the robust patch merging to reject outlier links, that is, the connectivity between two patches should be sufficiently redundant. However, this condition may not be satisfied in sparse subregions of a network. Thus, in such a sparse subregion, an outlier link may not be rejectable and will skew the location estimates of nearby sensor nodes. We address this challenge by proposing a robust patch merging operation that can detect the *non-rejectable outliers* and explicitly report their existence. When receiving such a report, a network localization algorithm can isolate the non-rejectable outliers. This network localization algorithm is called *RobustLoc* which, according to our proof, can reject outlier links reliably. RobustLoc has the advantage of rejecting outlier links reliably in sparse networks, compared with robust multilateration which can easily get stuck in sparse networks [Liu et al. 2005; Kung et al. 2009; Li et al. 2005; Kiyavash and Koushanfar 2007; Wang et al. 2007]. Our experiments show that in sparse networks with 5.5 degrees, RobustLoc can localize 90% of nodes, which strongly outperforms robust multilateration methods.

Besides outlier links, another key research issue we focus on is the threat of *outlier anchor nodes*. The anchor nodes are the sensor node with a priori knowledge of their locations. The locations of these nodes are defined in a global coordinate frame (GCF), for example, GPS coordinate frame. Their purpose is to guarantee that the locations of normal nodes are also defined in the GCF, which can be understood by the system users. However, it is inevitable to have outlier anchor nodes declaring erroneous locations that deviate from their true locations in GCF. The causes of outlier anchors can be misconfigurations when deploying the anchor nodes or malicious attacks, for example, replay attack and sybil attack [Newsome et al. 2004]. To make things worse, multiple outlier anchor nodes, due to malicious attacks, may collude to declare their locations in the same coordinate frame that is different from the GCF used by benign anchors.

The toleration of outlier anchors is a challenging task which is different from the toleration of outlier links, due to three reasons. (1) Outlier anchors may collude and declare positions in the same coordinate frame. In contrast, collusion among outlier links is rare, because it is difficult for the attackers to forge a set of outlier links that have an appropriate degree of deformation to let two patches merge consistently. Therefore, outlier anchors are more harmful than outlier links. (2) Outlier links may become non-rejectable due to the low connectivity issue which frequently occurs in sparse subregions of the networks. Such a phenomenon cannot be observed for outlier anchors. (3) The colluding outlier anchors can adopt a smart attacking strategy that seeks only local superiority, that is, the outlier anchors outnumber the benign anchors only in a small subregion. Thus, the sensor nodes in this small subregion may be

deceived to localize in the shared coordinate frame of the outlier anchors. The deceived sensor nodes can in turn help the outlier anchors deceive other nodes.

To reliably reject outlier anchors which may collude, we propose utilizing the *network-wide majority of benign anchors*, that is, benign anchors outnumber outlier anchors in the whole network region. To exploit such global knowledge, we propose an enhancement to the RobustLoc algorithm. This enhancement first constructs a largest local patch by an iterative local patch-merging process that excludes all the anchor declarations. Then, by merging this largest local patch with the global patch containing all the anchor declarations, we can reject colluding outlier anchors by *voting*. Such voting is implemented and encapsulated within our robust patch merging operation which reflects an elegant software design.

As a summary, this article makes the following key contributions.

(1) We propose the robust patch merging operation, which is more powerful and generalized than the traditional robust multilateration methods [Liu et al. 2005; Kiyavash and Koushanfar 2007]. Our RobustLoc algorithm, which invokes this new primitive operation, thus can effectively reject outlier links in sparse networks and meanwhile achieve high localization percentage. In contrast, robust multilateration methods can only localize a small proportion of nodes in sparse networks.
(2) Previous works neglect the situation that outlier links cannot be rejected, when the connectivity during patch merging is not sufficiently redundant. We have proposed an algorithm to detect and isolate such non-rejectable outlier links. This RobustLoc algorithm will be formally described in Algorithm II.
(3) We propose an enhancement to our RobustLoc algorithm to tolerate multiple outlier anchors which can collude due to malicious attacks. Our RobustLoc algorithm is the first to reject both outlier links and outlier anchors, as far as we know. In contrast, a recent paper [Jian et al. 2010] can only reject outlier links, which is based on the enumeration of realizable generic cycles.

These declared contributions will be validated by high-fidelity simulations with practical system parameters from Savvides et al. [2003] and Moore et al. [2004].

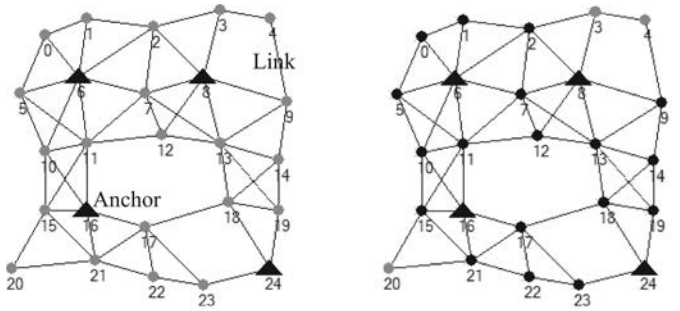
The rest of this article is organized as follows. Section 2 introduces the two primitive operations of network localization, that is, multilateration and patch merging. Section 3 presents our robust patch merging that can reject outlier links for these two primitive operations. Section 4 presents how we enhance the network localization algorithm to make it robust against outlier links. Section 5 addresses the problem of how to reject outlier anchors. Section 6 shows experimental results. Section 7 describes related work, and Section 8 concludes the article.

2. TWO PRIMITIVE OPERATIONS FOR NETWORK LOCALIZATION PROBLEM

In this section, we introduce the background knowledge for the network localization problem with an emphasis on the two following primitive operations to solve this problem.

(1) Multilateration that merges a sensor with a patch.
(2) Patch merging that merges two patches. Intuitively speaking, a patch is a group of nodes forming a rigid structure which is free from continuous deformation.

Network Localization Problem. The inputs of a network localization algorithm are a set of anchors with known locations and a set of link length measurements, as illustrated in Figure 2(a). The outputs of a network localization algorithm are the location estimates of the nodes that can be uniquely localized. These location estimates are depicted as black dots in Figure 2(b). The nodes that have ambiguous location assignments do not have black dots in Figure 2(b). For example, node 20 has ambiguous location



(a) Input: anchors and links. (b) Output: location estimates.

Fig. 2. Input and output of network localization problem.

(a) Iterative invocation of trilateration operation. Patch {A,B,C } expands to {A, B, C, D } then to {A, B, C, D, E }.

(b) Trilateration operation as a primitive.

Fig. 3. Construction of trilateration graphs by iteratively invoking the trilateration primitive.

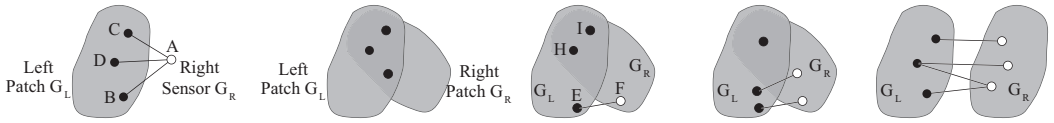
estimates and thus has no black dot, because this node can be flipped across the line through nodes 15, 21 without changing the length of the two links [20, 15] and [20, 21].

To solve this network localization problem, researchers investigated how a group of nodes should be interconnected to form a rigid structure. Such a rigid structure is free from any deformations (i.e., either flip or flex), and it can be localized if it contains three non-collinear anchors to fix it on a plane. Such a rigid structure is called a *generically globally rigid graph* [Eren et al. 2004] (or globally rigid graph for short), which is described in Theorem 2.1.

THEOREM 2.1 (GLOBALLY RIGIDITY & GENERICALLY RIGIDITY). *Graph G with at least four vertices is globally rigid in a two-dimensional space, if G is 3-connected and redundantly rigid. Graph G is redundantly rigid if G is generically rigid upon the removal of any edge. Graph G is generically rigid if it can satisfy Laman's Theorem. Please refer to Eren et al. [2004] for detailed discussions and proofs.*

Theorem 2.1 gives us a method for testing whether a subgraph is globally rigid. However, it is NP-hard to find all the globally rigid subgraphs in a topology and test whether they can be localized [Eren et al. 2004]. Therefore, people construct only a part of globally rigid patches by expanding the existing patches iteratively. A method of implementing such a patch expansion is to merge nodes into a patch iteratively, called *iterative trilateration*, which has polynomial computational cost [Savvides et al. 2003]. An example is in Figure 3. We start from the simplest rigid structures, that is, triangle $\{A, B, C\}$. We expand this rigid structure (called a patch) by merging node D into it, since this node is connected with the patch $\{A, B, C\}$ by three links. The operation that merges a node into a patch is called trilateration. By applying this trilateration operation iteratively, we can keep adding new nodes into the patch. The patches expanded in this way are called *trilateration graphs* which are guaranteed to be globally rigid [Eren et al. 2004].

However, iterative trilateration can easily get stuck in sparse networks with low connectivity and with sparse anchor distribution, for example, the network in Figure 2(a).



(a) Sensor merging by trilateration. **(b) Patch merging I.** **(c) Patch merging II.** **(d) Patch merging III.** **(e) Patch merging IV.**

Fig. 4. Globally rigid merging cases. (a) merges patch G_L with sensor G_R. (b)–(e) merge two patches G_L,G_R.

Therefore, recent work [Horn et al. 1988; Wang et al. 2008] considers not only the trilateration operation that merges a node into a patch but also the patch merging operation that merges two patches. We depict the patch merging operation in Figures 4(b)–(e) and describe it in Definition 2.2, which is similar to the theorems in Wang et al. [2008].

Definition 2.2 (*Globally Rigid Patch Merging*). The left patch G_L and the right patch G_R can be merged to generate a larger patch. The resultant patch is globally rigid if the two patches G_L and G_R are both globally rigid and can satisfy one of the following conditions.

—The two patches share three nodes, as in Figure 4(b).
—The two patches share two nodes and have one connecting link, as in Figure 4(c).
—The two patches share one node and have two connecting links, as in Figure 4(d).
—The two patches have four connecting links, as in Figure 4(e).

Moreover, for all these shared nodes and links, they should be relevant to at least three nodes in left patch G_L and relevant to at least three nodes in right patch G_R. Because the resultant patch is globally rigid, we call this merging a *globally rigid merging*.

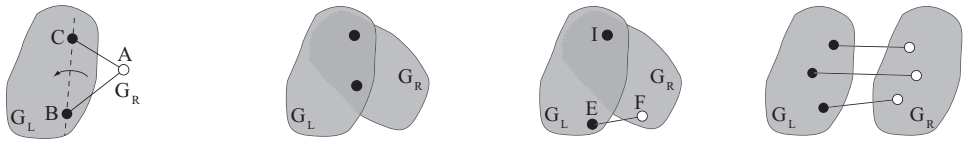
We can discover more globally rigid subgraphs in sparse networks by using patch merging plus trilateration than by using trilateration alone: for trilateration in Figure 4(a), we can use only the one-hop information in node *A*'s neighborhood, while for patch merging in Figures 4(b)–(e), we can exploit multi-hop information to align two patches.

To further increase the ability of discovering globally rigid structures (e.g., wheel graphs), a patch can be constructed with finite realizations, which is called a *generically rigid patch*. For example, the SWEEPS algorithm can merge a sensor node into a patch with only two links connecting them [Goldenberg et al. 2006]. This node however has two possible realizations, as shown in Figure 5(a), by flipping across the line through nodes B and C. The patches expanded with finite realizations are generically rigid but not globally rigid. We thus call the merging generically rigid merging. Besides the ambiguous merging of a node into a patch, two patches can also be merged with finite realizations [Wang et al. 2008]. We describe this generically rigid patch merging in Definition 2.3 and illustrate all the cases in Figures 5(b)–(d).

Definition 2.3 (*Generically Rigid Patch Merging*). The left patch G_L and the right patch G_R can be merged together to generate a larger patch. This resultant patch is generically rigid if the two patches G_L and G_R are generically rigid and they can fulfill one of the following conditions.

—The two patches share two nodes, as in Figure 5(b).
—The two patches share one node and have one connecting link, as in Figure 5(c).
—The two patches have at least three connecting links, as in Figure 5(d).

Moreover, for all these shared nodes and links, they should be relevant to at least two nodes in left patch G_L and relevant to at least two nodes in right patch G_R. Because the resultant patch is generically rigid, we call this merging generically rigid merging.



(a) Merge a sensor to a patch. (b) Merge two patches. (c) Merge two patches. (d) Merge two patches.

Fig. 5. Generically rigid merging cases. (a) merges patch G_L with sensor G_R. (b)–(d) merge patches G_L, G_R.

An Example of Iterative Localization. We explain how we use the globally rigid merging and the generically rigid merging to localize the sparse network in Figure 2(a). Triangle $\{2, 6, 7\}$ can be merged with triangle $\{6, 7, 11\}$ since they share two nodes, which corresponds to the case in Figure 5(b). The resultant patch $\{2, 6, 7, 11\}$ is generically rigid, and this patch can be merged with triangle $\{7, 8, 12\}$ because they share node 7 and have two links $[2, 8]$ and $[11, 12]$. This globally rigid merging generates a patch $\{2, 6, 7, 8, 11, 12\}$ which is a six-node wheel graph and is globally rigid. This patch and the global patch (containing all the anchors $\{6, 8, 16, 18\}$) share two nodes $6, 8$ and has one link $[11, 16]$, which satisfies the globally rigid merging condition in Figure 4(c). Thus, the nodes $2, 7, 11, 12$ can be merged into the global patch and be uniquely localized as shown in Figure 2(b). Other black nodes can be localized similarly by iterative patch merging.

In summary, for sensor network localization, there are two primitive operations to expand patches: multilateration as shown in Figures 4(a) and 5(a), and patch merging, as shown in Figures 4(b)–(e) and 5(b)–(d). Afterwards, we only use one term "patch merging", because multilateration can be regarded as a special case of patch merging if we regard an individual node (e.g., node A in Figure 4(a)) as a special one-node patch. For patch merging, we use the following symbols throughout this article.

G_L	the left patch during patch merging
G_R	the right patch (which may be just a sensor as special cases) during patch merging
$\mathcal{L}$	the set of links connecting left patch G_L and right patch G_R. Note that a node shared by the two patches G_L and G_R is contained in link set $\mathcal{L}$ as a zero-length link.

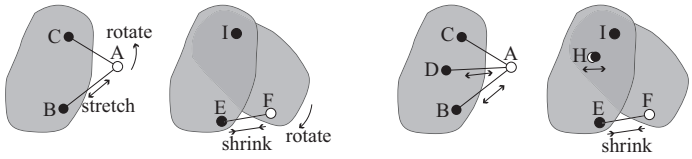
3. ROBUST PATCH MERGING OPERATION

In this section, we present the robust patch merging that is still accurate in the presence of outlier links whose length measurements severely deviate from their true values. This robust patch merging is important since the patch merging (with multilateration as a special case) is the primitive operation for network localization. In Section 3.1, we discuss the detection of outlier links during patch merging. When outlier links are detected, we need to identify and remove these outliers, which is addressed in Section 3.2. Finally, Section 3.3 presents reliable outlier link rejection against collusion.

3.1. Outlier Link Detection During Patch Merging

An outlier link can be intuitively understood as a link whose distance measurement error (i.e., the difference between the true distance and the measured distance) is abnormally large. For such outlier links, we provide a formal definition which assumes a ranging noise model based on Gaussian distribution.

Definition 3.1 (*Outlier Links Relative to Ranging Noise*). The links can be divided into two categories: normal links and outlier links. We assume that for normal links, their distance measurement errors exhibit the normal distribution $\mathcal{N}(0, \sigma)$, where σ is the ranging noise. This means that for 99.7% of all normal links, the absolute values of their measurement errors are within 3σ. Thus, we can define outlier links as the



(a) Non-detectable outlier links for generically rigid merging.

(b) Detectable outlier links for globally rigid merging.

Fig. 6. Outlier detectability of patch merging operations.

links whose distance measurement errors are abnormally large, for example, larger than 3σ. Such outlier link definition can be adjusted according to system requirement and distance measurement accuracy.

The preceding description can be translated into the following mathematical equation: for a link $\ell \in \mathcal{L}$, whether it is an outlier depends on the proposition $|err_\ell| > 3\sigma$, where err_ℓ is the distance measurement error with $err_\ell = d - \hat{d}$. Here, d is the true distance of link ℓ, and $\hat{d}$ is its measured distance by a certain ranging technique.

The outlier links may exist in link set $\mathcal{L}$ during patch merging. We argue that these outliers can be detected if the patch merging is globally rigid, which is depicted in Figure 4. We describe this condition for outlier link detection in Theorem 3.2.

THEOREM 3.2 (OUTLIER DETECTABILITY DURING PATCH MERGING). *Assume an outlier exists in link set $\mathcal{L}$ when left patch G_L is merged with right patch G_R. This outlier can be detected if the merging is globally rigid, as shown in Figure 4. This outlier cannot be detected if the merging is generically rigid but not globally rigid, as shown in Figure 5.*

PROOF. If the merging is barely generically rigid, we cannot detect outliers, since the removal of link ℓ ($\in \mathcal{L}$) makes some part of the graph flexible. For example, in Figure 6(a), if link $[A, B]$ is removed, node A can rotate around node C, and thus link $[A, B]$ if put back can deform freely. In Figure 6(a), link $[E, F]$ can also deform freely with the right patch rotating around node I.

The globally rigid merging topology, upon the removal of a nonzero-length link in $\mathcal{L}$, is still a generically rigid merging. Hence, for globally rigid merging, the deformation of outliers links incurs the deformation of normal links, which can be detected when we try to align the two patches G_L and G_R. For example, in Figure 6(b), the stretch of link $[A, B]$ incurs the stretch of link $[A, D]$, which is detectable; the shrink of link $[E, F]$ incurs the stretch of link $[H, H]$, if regarding shared node H as a zero-length link. □

Outlier Detection Implementation. When merging two patches G_L and G_R, we can detect outlier links by checking whether the set $\mathcal{L}$ contains any links with abnormally large deformations (or residuals). Residual calculation is a well-known concept in the localization community [Li et al. 2005; Kung et al. 2009; Jian et al. 2010]. We formulate this concept briefly. For a link $\ell \in \mathcal{L}$, its residual is calculated as

$$rsd_\ell = \hat{d} - |p - p^*|, \quad \text{where} \tag{1}$$

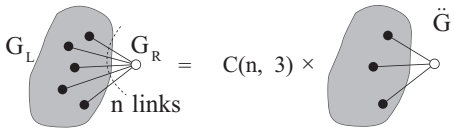
$\hat{d}$ is the measured distance for this link,

p is the location of one end of link ℓ after the patch merging, and

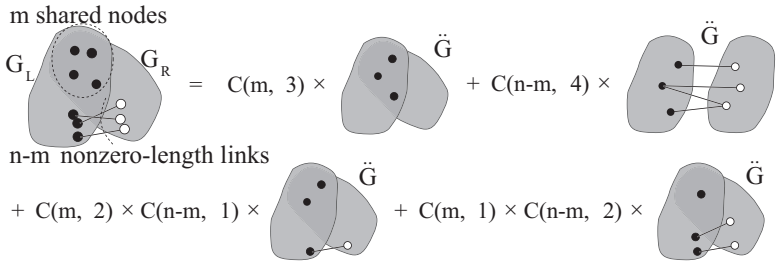
p^* is the location of the other end of link ℓ.

Thus, whether an outlier link can be detected depends on the following proposition

$$\forall\, \ell \in \mathcal{L}, \quad |rsd_\ell| > 2\,c\,\sigma, \quad \text{where } c \text{ is a constant configured as 3.} \tag{2}$$



(a) Enumerate minimally globally rigid merging subgraphs for multilateration.



(b) Enumerate minimally globally rigid merging subgraphs for patch merging.

Fig. 7. Enumerate minimally globally rigid merging subgraphs for the original merging topologies.

This equation checks the residuals of the links in set $\mathcal{L}$. But in practice, we check the residuals of the links both in set $\mathcal{L}$ and in two patches G_L,G_R, whose purpose is to detect and isolate the outlier links hiding in G_L,G_R. We will provide more details about this issue in Section 4.3. In summary, we can detect the outlier links by checking residuals.

3.2. Outlier Link Rejection During Patch Merging

Upon the detection of outlier links, we further need to identify which link or links are the outliers and remove them. The basic idea is to find a subset of link set $\mathcal{L}$ that can let the two patches G_L and G_R merge without detectable outliers. This subset of links are trusted to be normal links, and the other links in set $\mathcal{L}$ are identified as outlier links.

We present in Theorem 3.3 the necessary condition for outlier rejection during patch merging, that is, there exists a subset of links that can satisfy the globally rigid merging condition and have no detectable outlier links.

THEOREM 3.3 (OUTLIER REJECTABILITY DURING GLOBALLY RIGID PATCH MERGING). *When left patch G_L is merged with right patch G_R, outliers may exist in link set $\mathcal{L}$. These outliers can be identified only if the merging of G_L and G_R is still globally rigid upon the removal of outliers in link set $\mathcal{L}$.*

PROOF. If the merging upon the removal of outliers becomes generically rigid merging, then this subgraph with outliers removed may not be outlier-free by Theorem 3.2. Thus we cannot tell whether we have rejected all the outliers. □

According to Theorem 3.3, we need to find an outlier-free subgraph of the original patch merging topology that is globally rigid. However, if we check each globally rigid subgraph on whether it contains outliers, the computational cost can be as high as $O(2^n)$, where n is the number of links connecting the two patches G_L and G_R. We thus only check each subgraph that is minimally globally rigid. This concept of *minimally globally rigid merging subgraph* is defined in Definition 3.4 and illustrated in Figure 7.

Definition 3.4 (*Minimally Globally Rigid Merging Subgraph*). A patch merging topology G is comprised of left patch G_L, right patch G_R, and link set $\mathcal{L}$ with n links (see Figure 7). Given a merging topology G that is globally rigid, its subgraph $\ddot{G}$ is a minimally globally rigid merging subgraph if $\ddot{G}$ can satisfy all the following conditions.

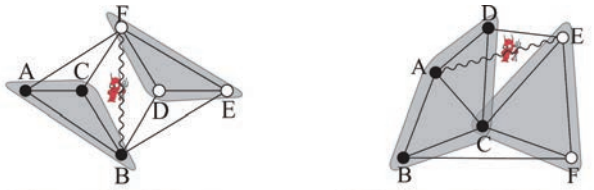


Fig. 8. Reject outliers by finding a minimally globally rigid merging subgraph without detectable outliers.

—$\ddot{G}$ contains the left patch G_L and the right patch G_R.
—the links in $\ddot{G}$ that connects G_L and G_R are a subset of $\mathcal{L}$ that connects G_L and G_R in the original merging topology G.
—$\ddot{G}$ can satisfy the globally rigid merging condition.
—$\ddot{G}$ cannot satisfy the globally rigid merging condition upon the removal of any link that connects the two patches G_L and G_R.

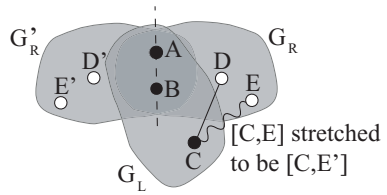
We illustrate in Figure 7 the enumeration of minimally globally rigid merging subgraphs for both the multilateration topology and the patch merging topology. Figure 7(a) depicts the subgraph enumeration for multilateration, and there are $C(n, 3)$ such subgraphs where n is the number of links. The studies in the field of robust multilateration essentially adopt the similar idea, because they reject outliers by enumerating the groups of three links [Kiyavash and Koushanfar 2007; Wang et al. 2007]. However, this method is just a special case of our subgraph enumeration method which can also be applied to patch merging, as illustrated in Figure 7(b). When there are m shared nodes during patch merging in Figure 4(b), we enumerate $C(m, 3)$ subgraphs that can barely satisfy the globally rigid merging condition. Since there are $n-m$ nonzero-length links, we further enumerate $C(m, 2) \times C(n-m, 1)$ subgraphs that can barely satisfy the condition in Figure 4(c). Similarly, there are $C(m, 1) \times C(n-m, 2)$ subgraphs that can barely satisfy the condition in Figure 4(d), and there are $C(n-m, 4)$ subgraphs that can barely satisfy the condition in Figure 4(e).

Examples of Outlier Rejection by Subgraph Enumeration. We exemplify how we apply the subgraph enumeration method to identify outliers during patch merging.

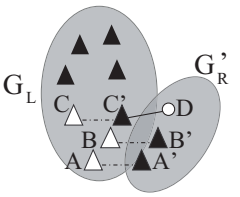
a. As illustrated in Figure 8(a), we can identify link $[B, F]$ as an outlier link when merging patch $\{A, B, C\}$ with patch $\{D, E, F\}$, because these two patches can be merged without detectable outliers if using the four links $[A, F]$, $[C, F]$, $[B, D]$, and $[B, E]$, which can satisfy the globally rigid merging condition depicted in Figure 4(e).
b. As illustrated in Figure 8(b), we can identify link $[A, E]$ as an outlier link when merging generically rigid patch $\{A, B, C, D\}$ with patch $\{C, E, F\}$, because these two patches can be merged without detectable outliers if using shared node C and two links $[B, F]$, $[D, E]$, which follows the patch merging case in Figure 4(d).

With the preceding examples, a possible outlier rejection method could work by two steps (1) Enumerate minimally globally rigid merging subgraphs, as shown in Figure 7 and check whether they have detectable outliers; (2) whenever a subgraph $\ddot{G}$ is found without detectable outliers, stop the enumeration and declare all the links that are consistent with $\ddot{G}$ to be normal links. Link-subgraph consistency relation is defined in Definition 3.5 such that the subgraph with the link added has no detectable outliers.

Definition 3.5 (*Link-Subgraph Consistency*). Given a globally rigid merging subgraph $\ddot{G}$, as in Definition 3.4, a link ℓ in link set $\mathcal{L}$ is consistent with $\ddot{G}$ if (1) $\ddot{G}$ has no detectable outliers and (2) $\ddot{G}$ with link ℓ added has no detectable outliers.



(a) Outlier link $[C,E]$ colludes with shared nodes A, B.



(b) Colluding outlier anchors A, B, C that declares positions A, B, C' in the same coordinate frame.

Fig. 9. Colluding outliers to justify voting algorithm. (a) Outlier link $[C, E]$ are non-rejectable due to flip ambiguity G'_R. (b) Outlier anchors A, B, C are non-detectable.

We note this consistency relation as $\ell \triangleright \ddot{G}$. By this definition, link ℓ is automatically consistent with $\ddot{G}$, if ℓ is contained in $\ddot{G}$.

3.3. Reliable Outlier Link Rejection Against Collusion

The outlier rejection method stated in the previous section can fail in the following two situations where there is collusion.

(1) Existence of an outlier link that colludes with normal links due to geometry effects. In such a situation, we can find a minimally globally rigid subgraph, that contains the outlier link and the normal links that collude with it. In this subgraph, we cannot detect the outlier link. For example, in Figure 9(a), link $[C, E]$ is an outlier link declaring its length equal to $[C, E']$. Then we can find a minimally globally rigid subgraph that contains shared nodes A, B and outlier link $[C, E]$. In this subgraph, we cannot detect the outlier since there are no abnormal deformations if the right patch is realized to G'_R.
(2) Existence of multiple outliers that collude with each other. We thus may find a minimally globally rigid subgraph with all its links to be colluding outliers, and we cannot detect them. For example, in Figure 9(b), three anchors A, B, C are outliers which declare their positions to be A', B', C' in the same coordinate frame. We cannot detect the outlier anchors because there are no abnormal deformations when the right patch is realized to G'_R. We will discuss the toleration of colluding outlier anchors in Section 5.

We present the pseudocode of our robust patch merging operation to detect and reject outliers in Algorithm I. This operation, if it cannot reject the outliers, can report such inability explicitly. To beat collusion, our basic idea is to use voting to exploit the majority of normal links over outlier links. There are four steps.

(1) Enumerate minimally globally rigid merging subgraphs and check whether they have detectable outliers.
(2) When a subgraph has no detectable outliers, calculate the votes that this subgraph can earn, and this subgraph can get the vote from a link if this link is consistent with this subgraph as defined in Definition 3.5.
(3) The subgraph with the highest votes wins, because the subgraph with only normal links will be supported by all normal links, which outnumber outlier links.
(4) When multiple subgraphs get the same highest votes, we can identify outliers if these subgraphs have the same set of supporters which is regarded as normal links. Otherwise, we report the inability to identity the outliers since there are multiple subgraphs getting the same highest votes but with different sets of supporters.

ALGORITHM I: Robust Patch Merging Based on Subgraph Enumeration and Voting

Input: A patch merging topology G which is comprised of left patch G_L, right patch G_R, and a set $\mathcal{L}$ of links between G_L and G_R,
Output: A larger resultant patch by merging G_L and G_R.

Procedure:
perform the merging of two patches G_L and G_R which are connected by link set $\mathcal{L}$
if *outliers can not be detected*, **then return** the merged patch directly

/* activate outlier rejection functionality */
initialize the maximum votes $v_{\max}$ as 0 and the winner set W as $\emptyset$
foreach subgraph $\ddot{G}$ of input topology G as shown in Figure 7 // subgraph enumeration
1. **if** $\ddot{G}$ *is not globally rigid merging*, **then continue**
2. **if** $\ddot{G}$ *is detected to contain outliers*, **then continue** // outlier detection
3. for candidate $\ddot{G}$, calculate its votes v obtained from the set of voters $\mathcal{L}$ as $v = \sum_{\ell \in L} \ell \triangleright \ddot{G}$, where $\ell \triangleright \ddot{G}$ means the link ℓ ($\in \mathcal{L}$) is consistent with the subgraph $\ddot{G}$ // voting
4. **if** $v < v_{\max}$, **then continue**
5. **if** $v > v_{\max}$, **then** $v_{\max} = v$ and reset W to empty
6. add subgraph $\ddot{G}$ to the winner set W // find the winners getting the highest vote

if *the winner set W is empty*, **then return** an empty resultant patch
if *the winners in W have different set of supporters*, **then return** an empty resultant patch
else regard the shared set of supporters for W as normal links, remove the identified outliers†, use only normal links to merge G_L and G_R, and **return** the merged patch

† Removal of a shared node or a zero-length link is to remove all the edges incident to the shared node.

Algorithm I can explicitly report its inability to reject the detected outlier. For example, in Figure 9(a), the outlier link $[C, E]$ colludes with normal links due to its special geometry. We cannot reject outlier link $[C, E]$, because the two realizations G_R and G'_R get the same votes (i.e., three votes) but have different sets of supporters. G_R is supported by shared nodes A, B and normal link $[C, D]$. G'_R is supported by shared nodes A, B and outlier link $[C, E]$. In this case, Algorithm I will return an empty resultant patch to indicate the failure of patch merging.

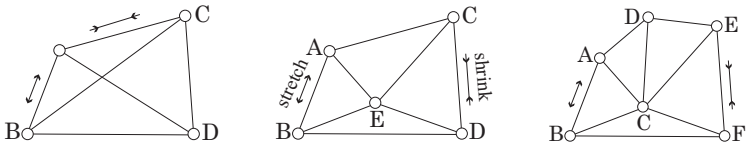
4. ROBUST NETWORK LOCALIZATION AGAINST NON-REJECTABLE OUTLIER LINKS

In this section, we solve the problem of robustly localizing a network in the presence of outlier links. At the present stage, we assume the absence of outlier anchors. Our basic idea of rejecting outlier links is to localize a network by iteratively invoking the robust patch merging operation in Algorithm I. However, the challenge is that the robust patch merging operation may sometimes fail to reject the detected outlier links, either due to collusion or due to insufficient connectivity, which is to be elaborated upon in Section 4.3. This challenge will be addressed in Section 4.2 by recording the failure of rejection and by isolating the non-rejectable outlier links.

4.1. Non-Rejectable Outlier Links in Patch Merging

The challenge to achieving robust network localization is that sometimes when merging two patches, we can detect outlier links but cannot reject them. This often happens in sparse subregions of a network containing outlier links. Such sparse subregions are called minimally globally rigid patches, as illustrated in Figure 10 and as defined in Theorem 4.1. If we do not properly handle the non-rejectable outlier links in such sparse subregions, the final location estimates could be biased.

THEOREM 4.1 (MINIMALLY GLOBALLY RIGID PATCH & OUTLIER LINK REJECTABILITY). *A patch G is minimally globally rigid if G is globally rigid and G becomes no longer globally rigid*



(a) Quadrilateral. (b) Five-node wheel. (c) Six-node wheel.

Fig. 10. Non-rejectable outlier links in minimally globally rigid patches. In such patches, deformation of a link incurs deformation of another link. Thus we cannot tell which link is the outlier.

upon the removal of any of its edge. If a minimally globally rigid patch G contains an outlier link, then this outlier link can be detected but cannot be rejected.

PROOF. Intuitively, if patch G is minimally globally rigid, then G has the necessary redundancy in connectivity to detect outliers, but G is not redundant enough to identify them. This is because G becomes generically rigid if one of its edges is removed; for a generically rigid graph, if one of its edges is removed, some part of it is flexible. For example, if links $[A, B]$ and $[A, C]$ are removed in Figure 10(a), node A can rotate about node D. For these two links, the stretch of one leads to the shrink of the other. We thus can detect the outlier by abnormal deformations but cannot tell which one is an outlier. □

According to Theorem 4.1, if an outlier link exists during the merging of two patches G_L and G_R, and the resultant patch is minimally globally rigid, this outlier link cannot be rejected. Such two patches G_L and G_R with non-rejectable outliers are defined as incompatible patches in Definition 4.2. There exists another situation in which two patches are incompatible with a non-rejectable outlier link, that is, the collusion of the outlier link with normal links, as shown in Figure 9(a).

Definition 4.2 (*Patch Incompatibility Relation*). Two patches G_L and G_R are defined to be incompatible, if their merged resultant patch contains an outlier link that cannot be rejected.

We explain why we have to merge the depicted patch pairs rather than the other patch pairs in Figure 8 in order to reject the outlier links, because other patch pairs that can be merged globally rigidly are incompatible. In Figure 8(a), patch $\{A, B, C\}$ is incompatible with patch $\{A, C, F\}$ because their resultant patch $\{A, B, C, F\}$ is minimally globally rigid and contains the outlier link; patch $\{B, D, F\}$ is incompatible with patch $\{D, E, F\}$ for the same reason. So we don't use them. In Figure 8(b), patch $\{A, B, C\}$ is incompatible with patch $\{C, E, F\}$ because the resultant patch $\{A, B, C, E, F\}$ is minimally globally rigid and contains the outlier link; patch $\{A, C, D\}$ is incompatible with patch $\{A, D, E\}$ for the same reason.

Our robust patch merging operation in Algorithm I can report the patch incompatibility relation (by returning an empty resultant patch), for the following two reasons. (1) Algorithm I can discover patch incompatibility due to the lack of redundancy in connectivity shown in Figure 10 for two reasons. First, Algorithm I can detect the outlier link because the merged patch is minimally globally rigid and the merging of G_L,G_R is globally rigid merging. Second, Algorithm I cannot identify the outlier link because we cannot find an outlier-free subgraph that is globally rigid. Algorithm I thus reports such non-rejectability by returning an empty merged patch. (2) Algorithm I can discover patch incompatibility due to the collusion between outlier link with normal links, as shown in Figure 9(a), which is already explained in Section 3.3.

ALGORITHM II. Robust Localization of Sensor Network N

Input: a set of anchors A_N whose locations are already known, and a set of links E_N with measured distances.
Output: Location estimates of uniquely localizable nodes.

Procedure:
Phase 1. Divide the Network into Basic Patches, including
—a global patch containing all the anchors A_N,
—local patches, each of which is a triangle,
—local patches, each of which is a single node.

Phase 2. Merge Patches Iteratively.
foreach *patch pair that are not marked incompatible and can satisfy the generically rigid merging condition*†, **do**

1. Merge two patches G_L and G_R connected by link set $\mathcal{L}$ (i.e., invoke the robust patch merging operation in Algorithm I)
   —globally rigidly if G_L and G_R can satisfy the condition in Figure 4, or
   —generically rigidly if satisfying the condition in Figure 5,
   and get the larger resultant patch G.
2. **if** *the above merging fails*‡, **then** mark patch pair $[G_L, G_R]$ as an incompatible pair and **continue** this foreach loop.
3. **if** *either one of patch pair* $[G_L, G_R]$ *is the global patch*, **then** mark the resultant patch G as a global patch‡†; **else** mark the resultant patch G as a local patch.
4. Destroy the two patches G_L and G_R.

Only the nodes in the global patch can be localized in global coordinate frame. But the global patch may not be globally rigid. Some of its nodes may have unique positions, and others may have ambiguous positions, which is called partial localizability.

† NOTE: If two patches can satisfy globally rigid merging condition, they can also satisfy generically rigid merging condition, and they are given a higher priority to be merged than the patch pairs that cannot satisfy globally rigid merging condition.

‡ Failure to reject detected outliers is indicated by the empty patch G obtained at step 1 of phase 2.

‡† If the resultant patch is merged from a local patch and a global patch, it should contain at least three anchors. A generically rigid structure with at least three anchors is a global patch that can be localized ambiguously.

4.2. Robust Network Localization

We present in Algorithm II our network localization algorithm based on iterative patch merging. This algorithm is robust with the presence with outlier links, because it (i) removes the rejectable outlier links by robust patch merging operation (i.e., Algorithm I) and (ii) isolates the non-rejectable outlier links when robust patch merging operation reports patch incompatibility relation.

Algorithm II is composed of the following two phases.

Phase 1 divides the network into basic patches, including a global patch and many local patches. Each local patch can be either a triangle or an individual node.

Phase 2 merges these patches iteratively and each local patch that is merged into the global patch can be localized finally. All cases for two patches to be merged are already shown in Figure 4 for globally rigid merging and in Figure 5 for generically rigid merging. At step 1 of phase 2, we merge two patches G_L and G_R by our robust patch merging operation in Algorithm I. This operation can reject outlier links. Several experiments of outlier link rejection can be found in Section 6.3.

Algorithm II can isolate non-rejectable outlier links, because when two patches are incompatible with non-rejectable outliers, the robust patch merging operation in Algorithm I can detect such a situation. Then at step 2 of phase 2, we memorize the detected incompatible patch pair, and afterwards we will never attempt to merge this incompatible patch pair, thanks to the guarding condition of the `foreach` loop of phase 2. Finally, Algorithm II can guarantee that the globally rigid subgraphs of the global patch are free from detectable outlier links. Only in these subgraphs can the nodes be localized and are immune to the adverse impact of outlier links.

4.3. Analysis of Robustness Against Outlier Links

Single Outlier Link. When a single outlier link exists during the merging of two patches, our Algorithm II can defend against (reject or isolate) this outlier, which is proved as follows.

PROOF. When merging two patches, there are two possibilities with regards to where the outlier links are contained. (1) The outlier links may be contained in the link set $\mathcal{L}$ that connects the two patches G_L and G_R. These outliers can be rejected by our robust patch merging algorithm in Algorithm I, which can find a trustworthy link subset excluding these outliers, as shown in Figure 8. (2) The outlier links may also exist within the patch G_L or patch G_R. The patches that contain outlier links are called ill patches. We prove as follows that the ill patches can be isolated by our Algorithm II.

For an ill patch, there are only two possible cases.

—The ill patch is globally rigid. However, this case is impossible, because the globally rigid patches have the necessary redundancy to detect the outlier links. Such a detection can be implemented by checking whether the residuals of all the links are within a threshold. If a patch is detected to be ill, we can remove it for safety's sake.
—The ill patch is generically rigid but not globally rigid. This case is possible because such patches do not have the necessary redundancy in connectivity to detect the outlier links. For example, a triangle may contain an outlier link, and this outlier can not be detectable, because each of the triangle edges can deform freely.

We can isolate the ill patches containing outlier links, because the outlier links can strongly distort node positions in the ill patch. When this patch is merged with a healthy patch that is free from outliers, our Algorithm II will record the two patches to be incompatible (i.e., the algorithm can detect the presence of strong deformations in the links connecting the two patches, no matter which subset of links $\mathcal{L}$ we use). For the similar reason, this ill patch will be recorded to be incompatible with other healthy patches and get isolated finally. For algorithm details, please check out the descriptions of Algorithm II in Section 4.2. □

All Links are Outliers. When all the links are outliers during patch merging, our robust patch merging algorithm (i.e., Algorithm I) cannot reject these outliers. Neither can the conventional method based on robust multilateration.

Algorithm I cannot reject these outliers because our subgraph enumeration method (as shown in Figure 7) cannot find a subset of links that can let the two patches merge without detectable outliers.

However, we argue that, in such situations, Algorithm I can detect these outlier links, because it can find that the residuals of some links are above a threshold (i.e., Theorem 3.2). Thus, Algorithm I can explicitly report its inability to reject the detected outliers. When this report is received by the robust network localization algorithm (i.e., Algorithm II), this algorithm will avoid merging these two patches, which is essentially a fail-safe procedure to make the iterative patch merging safe. In summary, when all

the links are in error, our Algorithm II cannot precisely reject these outlier links but can isolate these outlier links from normal links.

All Links are Colluding Outliers. Another question is what if all the links are outliers and they collude to let the two patches merge without detectable outliers. Both our robust patch merging (i.e., Algorithm I) and the conventional robust multilateration will fail in the outlier detection. However, it is difficult, if not impossible, for the attackers to forge such a set of colluding outlier links. It is more convenient for the attackers to forge multiple colluding anchors which declare erroneous positions in a shared coordinate frame, which will be discussed in the next section.

5. ROBUST NETWORK LOCALIZATION AGAINST OUTLIER ANCHORS

In this section, we solve the problem of robustly localizing a network in the presence of outlier anchors. In Section 5.1, we expose an inadequacy of Algorithm II, that is, this robust network localization algorithm cannot reliably reject outlier anchors, even when there is only one outlier anchor in the network. In Section 5.2, we present an enhancement to Algorithm II to reliably reject outlier anchors. These multiple outlier anchors can collude to declare their positions in the same coordinate frame. Finally, we analyze the robustness of this proposed RobustLoc algorithm against various malicious attacks relating to outlier anchors.

5.1. Rejection of Single Outlier Anchor

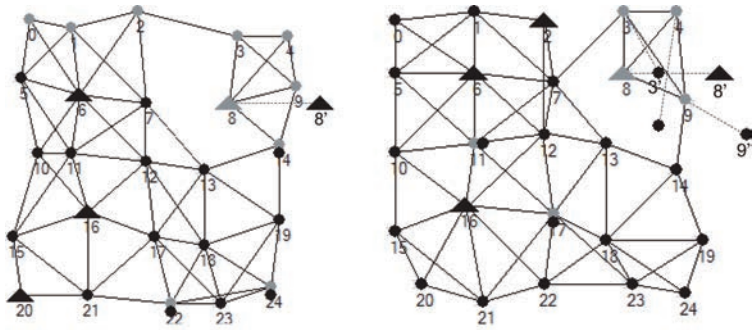
An outlier anchors is an anchor node which declares its position in an erroneous coordinate frame that is different from the global coordinate frame (GCF). The causes of outlier anchors can be misconfigurations when deploying the anchor nodes or malicious attacks. We provide a formal definition for outlier anchors as follows.

Definition 5.1 (*Outlier Anchors Relative to Ranging Noise*). The anchor nodes can be divided into two categories: normal anchors and outlier anchor. We assume that for normal anchors, their positioning errors are much smaller than ranging noise σ to guarantee the final localization accuracy. We define outlier anchors as the anchor whose positioning errors are abnormally larges than ranging noise, for example, larger than 3σ. Such outlier anchor definition can be adjusted according to system requirement and ranging noise.

The network localization algorithm presented in Algorithm II in fact cannot reliably reject a single outlier anchor in a network. This inadequacy is not obvious because it sounds reasonable that an outlier anchor is equivalent to a normal anchor whose incident links are all outlier links. Then a robust network localization algorithm, if it can reject links with abnormal deformations, can remove the normal links incident to outlier anchors and thus automatically reject outlier anchors.

However, we point out that the removal of normal links incident to outlier anchors may cause two problems, even when there is only one outlier anchor in a network. We describe these two problems as follows and also illustrate them in Figure 11.

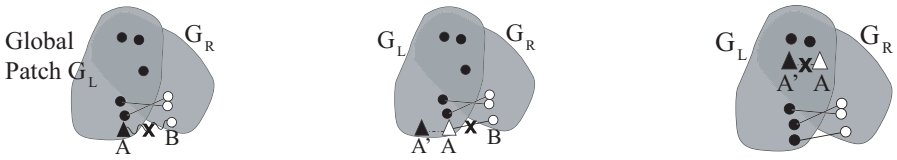
(1) *Reduced Localization Percentage*. In Figure 11(a), anchor 8 is an outlier. If the normal links [7, 8] and [13, 8] incident to the outlier anchor 8 are removed, patch $\{3, 4, 8, 9\}$ cannot be uniquely localized because it is connected to the network by just three links [4, 8], [4, 9], and [2, 3].
(2) *Localized to Wrong Realizations*. In Figure 11(b), with the removal of normal links [7, 8] and [13, 8], the patch $\{3, 4, 8, 9\}$ is localized to the depicted wrong realization $\{3', 4', 8', 9'\}$. Note that this wrong realization has no detectable outliers since it has no abnormally deformed links, that is, distance [7, 3] is equal to [7, 3′], and distance [14, 9] is equal to [14, 9′].



(a) Problem 1: patch $\{3, 4, 8, 9\}$ cannot be localized.

(b) Problem 2: Patch $\{3, 4, 8, 9\}$ is localized to a wrong realization.

Fig. 11. Two problems of incorrectly removing normal links [7, 8] and [13, 8] incident to outlier anchor 8.



(a) Reject outlier link $[A, B]$. (b) Falsely reject normal link $[A, B]$. (c) Reject outlier anchor A.

Fig. 12. Reliable rejection of single outlier anchor. In both (a) and (b), abnormal deformation can be found in link $[A, B]$. But we cannot tell whether this deformation is caused by outlier link $[A, B]$, as shown in (a), or by outlier anchor A, as shown in (b).

Considering these two harms, we need to reject outlier anchors precisely, rather than falsely reject several innocent normal links that are incident to the outlier anchors. We describe the condition of precise outlier anchor rejection as follows.

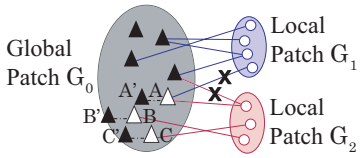
THEOREM 5.2 (PRECISE REJECTION OF AN OUTLIER ANCHOR DURING PATCH MERGING). *We need to merge left patch G_L with right patch G_R. Assume G_L is the global patch that contains all the anchor nodes, one of which is an outlier. During the merging, the condition for the precise rejection of this outlier anchor without the removal of its incident normal links is that the outlier anchor is a shared node between G_L and G_R, like the anchor A shown in Figure 12(c).*

PROOF. When merging global patch G_L with a local patch G_R, if the outlier anchor is only incident to a link connecting the two patches, as in Figure 12(b), then it is impossible to tell whether this abnormal deformation of link $[A, B]$ is caused an outlier anchor A, as in Figure 12(b) or by caused by an outlier link $[A, B]$, as in Figure 12(a). If we simply regard link $[A, B]$, as an outlier link, then the removal of normal links may cause two problems, as described before.

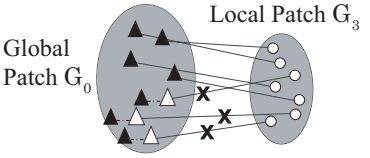
In contrast, if the outlier anchor is a shared node between two patches in Figure 12(c), we can know unambiguously that anchor declaration A' is erroneous, because A' is contained in global patch G_L, node A is contained in local patch G_R, and zero-length link $[A', A]$ is abnormally stretched. □

5.2. Rejection of Multiple Outlier Anchors

The rejection of multiple colluding outlier anchors is different from the rejection of a single outlier anchor, because *local knowledge* may not be enough to reject the colluding

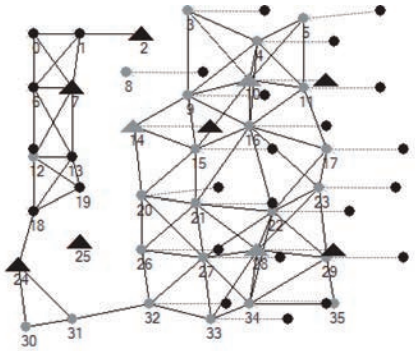


(a) Patches G_1, G_2 with local knowledge.

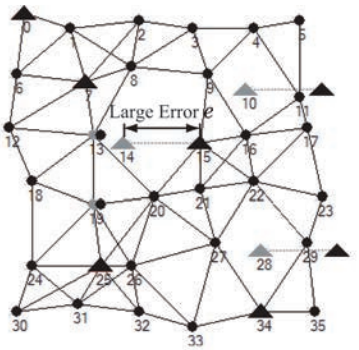


(b) Patch G_3 with global knowledge.

Fig. 13. Networks with multiple outlier anchors. Note that a link depicted in this figure represents a shared node if the link length is zero.



(a) Sensors are cheated by colluding outlier anchors.



(b) Successful rejection of colluding outlier anchors.

Fig. 14. Localization of networks with four normal anchors and three colluding outlier anchors 10, 14, 28. (a) is not configured with delayed global patch merging feature. (b) is configured with such a feature.

outlier anchors. Here, local knowledge means that not all the anchors are incident to the links $\mathcal{L}$ during the merging of the global patch G_L with a local patch G_R.

Illustration. We illustrate the potential harms that can be brought about by using only local knowledge to reject colluding outlier anchors. In Figure 13(a), global patch G_0 contains three colluding outlier anchors A, B, C which declare their positions to A', B', C' in the same coordinate frame. We can still correctly reject the outlier anchor A when we merge global patch G_0 with local patch G_1, because most of the links connecting them are incident to normal anchors rather than the outlier anchor A. According to Theorem 5.2, this single outlier anchor A can be reliably rejected if it is a shared node between the two patches G_0 and G_1. Note that it is possible for the links depicted in Figure 13 to be zero-length links that are also shared nodes. However, we cannot reliably reject the outlier anchors when we merge the global patch G_0 with local patch G_2, because among all the links connecting them, the links incident to the three outlier anchors are the majority and thus these outlier anchors can deceive the local patch G_2 to be localized in their coordinate frame.

Experiment. We verify by experiments that only the local patches with local knowledge can be deceived by colluding outlier anchors. In Figure 14(a) with three colluding outlier anchors 10, 14, 28; all the sensors in the right part of the network are shifted rightwards to be localized in the coordinate frame of these outlier anchors. In contrast, in the left part of the network where the normal anchors dominate, sensors are still localized in the correct coordinate frame.

Therefore, when there are multiple colluding outlier anchors, we need network-wide *global knowledge* to reliably reject them. The concept of global knowledge is formally defined in Definition 5.3, and it is illustrated in Figure 13(b), which means that the local patch G_3 has links incident to all the anchors within the network during its

merging with the global patch G_0. Patch G_3 thus has enough information to exploit the network-wide majority of normal anchors over outlier anchors. The voting method that can exploit this majority is already implemented within our robust patch merging algorithm in Algorithm I.

Definition 5.3 (*Global Knowledge for Reliable Rejection of Outlier Anchors*). Multiple colluding outlier anchors can be rejected reliably by our robust patch merging operation under the following conditions. (1) Given link set $\mathcal{L}$ connecting the global patch G_L and a local patch G_R, for each anchor in the global patch G_L, there is a link in $\mathcal{L}$ incident to it, as shown in Figure 13(b). (2) In the link set $\mathcal{L}$, the links incident to normal anchors are the majority.

Delayed Global Patch Merging. To reliably reject colluding outlier anchors, we propose an improvement (named *delayed global patch merging*) to the iterative patch merging process in Algorithm II. Its intuition is to make use of global knowledge to reject the colluding outlier anchors. Therefore, we need to construct a large local patch having global knowledge that can satisfy the condition in Definition 5.3. When merging this large local patch (with global knowledge) into the global patch, we can differentiate normal anchors from outlier anchors by the voting algorithm which is embedded within the robust patch merging operation in Algorithm I.

To construct such a large local patch with global knowledge, we merge the initially constructed small local patches iteratively with all the anchor nodes (i.e., the global patch) excluded from this iterative merging process. When this iterative local patch-merging process terminates, we find the largest local patch that remains, which is generically rigid. Then we merge this local patch with the global patch using our robust patch merging algorithm in Algorithm I. During this merging, all the anchors becomes shared nodes because of the large size of this largest local patch remained. This merging thus can reject the outlier anchors reliably according to Theorem 5.2. We name such a localization process with the local patches merged first without the global patch as our *RobustLoc* algorithm.

5.3. Security Analysis of RobustLoc Algorithm

We describe various attack models relating to outlier anchors as follows. Our analysis show that these attacks can be defended against if using our robust network localization algorithm (RobustLoc).

Single Compromised Anchor. We assume that each anchor has its unique cryptographic key; otherwise, the leak of the shared key would wreck the whole network. Thus, if an attacker has cracked the cryptographic key of an anchor, then a compromised anchor exists which declares an erroneous position by the will of the attacker. We can tolerate the single compromised anchor due to the leak of key, because, according to Theorem 5.2, this outlier anchor can be reliably rejected if this anchor is a shared node during patch merging, which is also illustrated in Figure 12(c). This condition can be satisfied by our enhancement called delayed global patch merging. In this enhancement, we first let the local patches merge without the anchor nodes. When a local patch grows large enough, it will contain the single compromised anchor. Thus, this compromised anchor will be a shared node when merging this large local patch with the global patch and can be rejected reliably.

Replay Attack and Sybil Attack. An attacker can launch a Replay attack, if it overhears an anchor declaration and replays this declaration at other places. The replay attacks can be defeated by Algorithm II, which is a centralized algorithm that assumes each anchor identity has only one position declaration (see the input anchor set A_N of Algorithm II). Thus, the duplicated position declarations with the same anchor identity

can be compressed to one declaration. This single outlier anchor can be tolerated by our Algorithm II, as described before. However, a safer solution is to revoke this compromised anchor identity upon the detection of a cloned anchor identity with multiple position declarations.

An attacker can launch a Sybil attack, if it grabs a compromised anchor, exploits this anchor's identity, and makes falsified anchor declarations at different places [Newsome et al. 2004]. A Sybil attack is different from a Replay attack, because the Sybil attacker can forge multiple anchor identities which can not be compressed or revoked. This Sybil attack is treated the same as the following *multiple compromised anchors attack*.

Multiple Compromised Anchors. This attack is the most troublesome since these multiple outlier anchors may collude to declare erroneous positions in the same coordinate frame. To make thing worse, the attacker may adopt a smart strategy that first seeks *local superiority* and finally achieves *whole-field superiority*. This means that the attacker understands that he cannot obtain compromised anchors with a sufficient quantity to outnumber the benign anchors in the whole field, so he controls the deployed positions of these compromised anchors to allow them to outnumber the benign anchors in a particular small region, which is called the local superiority. Then, the sensors in this small region will be deceived, since most of their distance measurements are incident to the compromised anchors rather than the benign anchors. These deceived sensors may in turn help the malicious anchors to deceive other sensors to localize in the erroneous coordinate frame. The worst situation is that finally, most of the nodes in the sensor field will be deceived by a few compromised anchors, which is called the whole-field superiority. Such a worst case has already been depicted in Figure 14(a).

To defeat this local-superiority strategy, we have proposed the enhancement called delayed global patch merging. That is, we first l et the local patches merge without the anchor nodes. When the local patches grow large enough, they will have contact with most of the anchor nodes in the network. This global knowledge can help the local patch to differentiate and reject the malicious anchors with only local superiority, which is illustrated in Figure 13(b). Our experiment in Figure 14(b) shows that our RobustLoc can reliably reject such multiple compromised anchors with local superiority.

6. EXPERIMENTAL EVALUATION

In this section, we verify the robustness of our RobustLoc algorithm against outlier links and outlier anchors in various network conditions. Section 6.1 briefly describes experimental settings. Section 6.2 verifies that our robust patch merging operation in Algorithm I can effectively detect and reject outlier links. Section 6.3 shows that our robust network localization algorithm RobustLoc can achieve higher localization percentage than RobustMultilateration [Kiyavash and Koushanfar 2007] in sparse networks. Section 6.4 shows that our RobustLoc algorithm is more robust and accurate than state-of-the-art localization algorithm CALL [Wang et al. 2008] by rejecting outlier links and outlier anchors. Finally in Section 6.5, we show that RobustLoc can provide good localization accuracy and high localization percentage in concave networks.

6.1. Experimental Settings

Our experiments assume the following system parameters. We assume Cricket for the ranging system, which is based on ultrasonic TOA. Thus the ranging noise σ is configured as 2 cm and the ranging radius r is set to 6 m, as listed in the following table. For the ranging noise model, we adopt the empirical model in Whitehouse et al. [2005] with heavy tails to overestimate distance probably due to non-line-of-sight conditions. For outlier distances and outlier anchors with abnormally large error, their error magnitude

e can be adjusted. For network topology, nodes are arranged by a disturbed grid. By adjusting the grid spacing s, we can generate networks with a different degree d.

Configured Parameters of Wireless Network Localization	
σ	expected noise of ranging methods, configured as 0.02m
r	radius to effectively obtain ranging data, configured as 6m
s	spacing of disturbed grid for node deployments
d	network degree (networks with $d < 7$ are sparse networks)
e	error magnitude of outlier anchors and outlier links

Compared Variables of Wireless Network Localization	
a	average localization accuracy (i.e., distance from true location to estimated location) of uniquely localized nodes
p	percentage of uniquely localizable nodes among all nodes

For the following three localization algorithms, we will compare their localization accuracy a and localization percentage p.

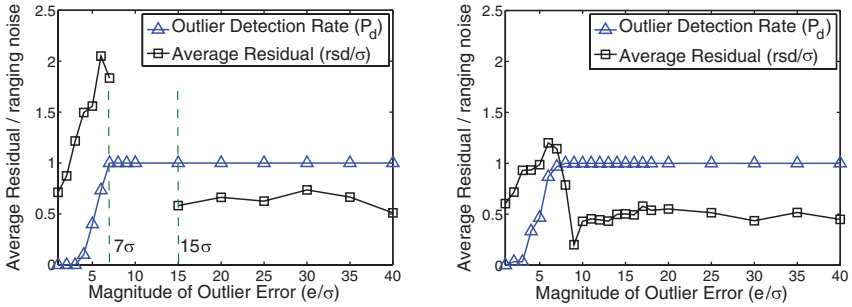
(1) *RobustLoc*. Our robust localization algorithm in Algorithm II that invokes the robust patch merging operation in Algorithm I and adopts the delayed global patch merging feature described in Section 5.2.
(2) *RobustMultilateration*. The iterative multilateration algorithm in [Savvides et al. 2003] that invokes the robust multilateration in Kiyavash and Koushanfar [2007].
(3) *CALL*. A localization algorithm that iteratively invokes the patch merging operation in Figures 4 and 5, without any enhancements of outlier rejection [Wang et al. 2008].

6.2. Impact of Ranging Noise to Outlier Rejection

In our first experiment, we do not investigate the robustness of our RobustLoc algorithm. Instead, we check the effectiveness of our primitive operation (i.e., the robust patch merging in Algorithm I) in detecting and rejecting outliers when the ranging noise σ is present. According to Definition 2.2, a link is an outlier links if its error is larger than 3σ. Ideally, it is best to reject all these outliers. However, the reality is that the outlier detection threshold is much larger than 3σ. This experiment shows Algorithm I can reliably reject or isolate the outlier links with nearly 100% success rate only when the outlier error e is larger than 7σ (for multilateration) or larger than 10σ (for patch merging). The possible causes are the geometry effect (which is also called geometric dilution of precision in GPS system) plus the error accumulation in recursive patch merging process. Overall, this experiment shows that we can keep the average link residual rsd below 2σ, no matter what the value for the outlier error.

Compared Variable for Patch Merging	
rsd_ℓ	Residual of link ℓ, that is, the difference between measured distance $\hat{d}$ and estimated distance $\lvert p - p^* \rvert$, where p is the position of one end of link ℓ after patch merging and p^* is for the other end.
P_d	The probability of detecting the outliers. We can detect the outliers if we find any link $\ell \in \mathcal{L}$ where its residual rsd_ℓ exceed the threshold $2c\sigma$.
rsd	Average residual of the remaining links with outliers rejected in link set $\mathcal{L}$.

Figure 15(a) shows that our robust patch merging operation can effectively detect and reject outliers for the multilateration topology depicted in Figure 7(a). The experimental topology has four normal links and one outlier link. Algorithm I can detect the existence of the outlier by checking whether the residual rsd_ℓ of any link exceeds the threshold $2c\sigma$, where c is a constant. In Figure 15(a), Algorithm I can detect outliers with nearly 100% success rate ($P_d \approx 1$) when the outlier error e is larger than 7σ. Although the outlier cannot be effectively detected when e is smaller than 7σ,



(a) Multilateration topology in Figure 7(a). (b) Patch merging topology in Figure 7(b).

Fig. 15. Robust patch merging operation vs. outlier error. (a) shows the rejection in multilateration topology with one outlier link. (b) shows the rejection in patch merging topology with one outlier link.

the average residual is still kept below 2σ. When e is larger than 15σ, Algorithm I can identify the outlier and reject it. Thus the average residual is around 0.5σ. When e is between 7σ and 15σ, Algorithm I reports that the outlier can be detected but cannot be identified. This is because Algorithm I can find multiple winners in the winner set W having the same set supporters.

Figure 15(b) shows that our robust patch merging operation can effectively detect and reject outliers for the patch merging topology, as illustrated in Figure 7(b). Outlier rejection in such a topology cannot be handled by RobustMultilateration. The experimental topology has five normal links and one outlier link. In Figure 15(b), the outlier detection rate P_d is nearly 100% when outlier error e is larger than 7σ. The average residual is kept below 1.5σ by rejecting the outlier.

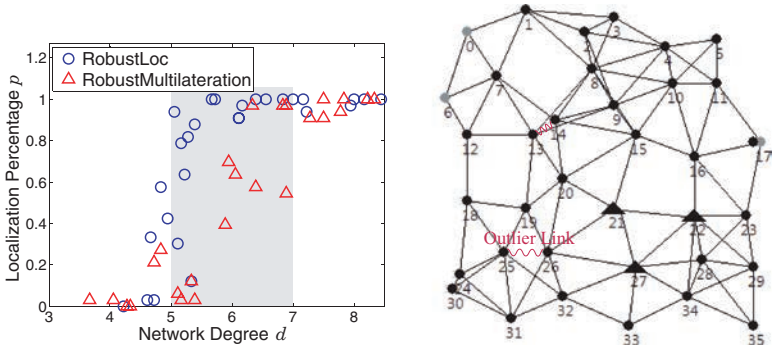
6.3. Outlier Links Toleration in Network Localization

This experiment shows that RobustLoc can effectively tolerate outlier links while achieving high localization percentages in sparse networks. Thus we compare the localization percentage p of RobustLoc and RobustMultilateration. This experiment assumes the presence of outlier links and the absence of outlier anchors in the simulated networks. We also assume that three anchors in the simulated network are geographically close (like anchors 21, 22, 27 in Figure 16(b)), since RobustMultilateration needs these nearby anchors to bootstrap the algorithm.

Figure 16(a) shows that in sparse networks with degrees between 5 and 7, the localization percentage of RobustLoc is higher than that of RobustMultilateration by at least 30%, because the localization percentage of RobustLoc is generally above 80% and that of RobustMultilateration is below 60%. Figure 16(b) depicts a network with a degree of 5.05 and with two outlier links [13, 14], [25, 26]. In such a sparse network, RobustMultilateration cannot localize any nodes because there does not exist a node with three links connecting to anchors 21, 22, 27. However, if using RobustLoc, the localization percentage is as good as 93%, and meanwhile, the location estimates (i.e., black dots) are close to their true positions.

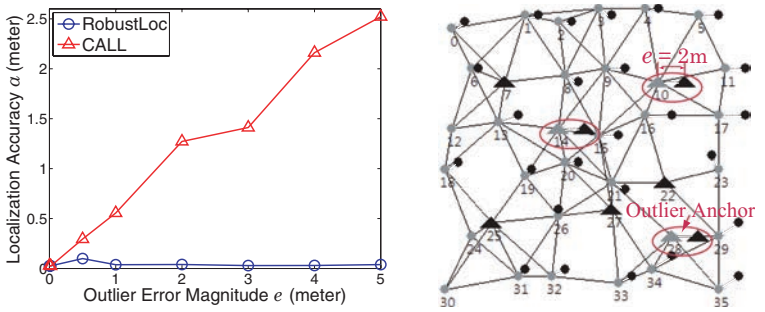
6.4. Toleration of Colluding Outlier Anchors in Network Localization

This experiment shows that RobustLoc can effectively tolerate colluding outlier anchors in sparse networks. We vary the outlier error magnitude e and compare localization accuracy a of RobustLoc and CALL. The simulated networks are sparse networks with degree d around 5.5. In these networks, we deploy seven anchors randomly and three of them are colluding outlier anchors, for example, 10, 14, 28 in Figure 17(b).



(a) Comparing localization percentage. (b) Localization percentage 93%.

Fig. 16. Comparison of RobustLoc and RobustMultilateration in sparse networks. (a) A comparison of localization percentage. (b) Localization result of RobustLoc in a sparse network with 5.05 degree and with outlier links [13, 14], [25, 26]. In this network, RobustMultilateration cannot bootstrap.



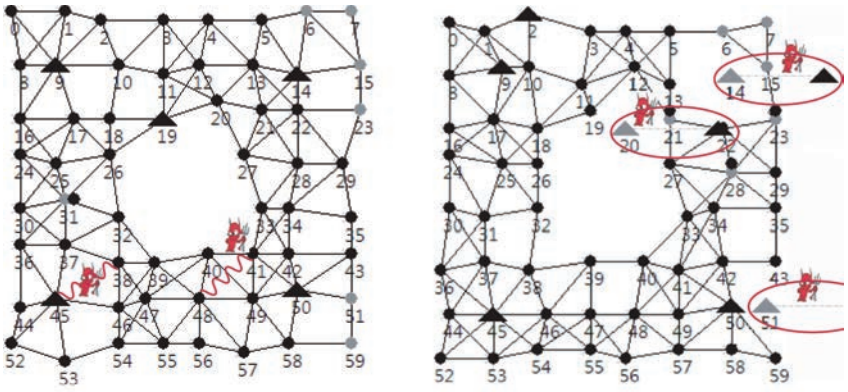
(a) Impact of outlier error on accuracy. (b) CALL with error e = 2m .

Fig. 17. RobustLoc vs. CALL in localization accuracy a.

Figure 17(a) shows that RobustLoc is robust against outlier anchors but CALL is not. This is because the localization accuracy a of RobustLoc is proportional to ranging noise σ regardless of error e of outlier anchors, but accuracy of CALL is proportional to error e. RobustLoc is robust against outlier anchors because RobustLoc can identify and isolate the three outlier anchors, as shown in Figure 14(b). CALL is not robust because CALL has no outlier rejection ability. Thus, all the sensor nodes shown in Figure 17(b) are dragged from their true positions by the outlier anchors. The magnitude of such deviations are proportional to error e of outlier anchors.

6.5. Concave Network Deployment Regions

This experiment verifies that RobustLoc can reliably reject outliers in concave networks. We simulate O-shaped networks as shown in Figure 18, which are sparse networks with a degree around 5. Although Figure 18(a) contains two outlier links [41, 48] and [38, 45], our RobustLoc algorithm can still provide good localization accuracy with average error below 2σ and satisfying localization percentage above 80%. Figure 18(b) depicts an O-shaped network with colluding outlier anchors 20, 14, 51. Our RobustLoc algorithm can effectively identify and isolate these outliers.



(a) Reject outlier links; LocError = 1.35σ; LocPercentage=89%.

(b) Reject outlier anchors; LocError = 1.46σ; LocPercentage=94%.

Fig. 18. Outlier rejection demo in O-shaped networks.

7. RELATED WORK

Network localization techniques can be divided into two major categories according to *ranging accuracy*.

(1) *Coarse-grained* techniques have low ranging accuracy of meters or tens of meters since they exploit radio attenuation for ranging [Li and Liu 2007; Lim and C. 2005; He et al. 2003; Wang and Xiao 2006; Xiao et al. 2010b; Xiao et al. 2010a; Mao and Fidan 2009]. They have two advantages. The first is their low cost with no requirement for extra ranging hardware. The second is their ability to satisfy accuracy requirements of certain location-dependent protocols and applications [He et al. 2003].
(2) *Fine-grained* techniques can provide high accuracy at the centimeter level, which is based on TOA techniques, for example, ultrasonic TOA or RF TOA based on ultra-wideband signals [Savvides et al. 2003; Goldenberg et al. 2006; Horn et al. 1988; Wang et al. 2008; Shang and Ruml 2004; Moore et al. 2004; Priyantha et al. 2003]. These techniques can achieve high ranging accuracy (often better than 1m) at the price of deploying a ranging device per node.

This article focuses on the fine-grained localization in which the internode distances can be accurately measured. From the distance measurements, node locations need to be derived. In this field, most of the solutions are based on two primitive operations, that is, multilateration and patch merging. The previous papers which discuss these two operations focus on different perspectives.

—The pre-conditions of invoking the primitive operations are discussed [Savvides et al. 2003; Horn et al. 1988; Goldenberg et al. 2006; Wang et al. 2008]. Their aim is to improve the percentage of localizable nodes in sparse networks.
—The toleration of ranging noises to improve the localization accuracy is investigated in [Foy 1976; Priyantha et al. 2003; Shang and Ruml 2004]. The common characteristic of these papers is the adoption of certain numerical optimization techniques.
—The robustness against patch flipping is studied [Moore et al. 2004; Kannan et al. 2011; Wang et al. 2010]. Patch flipping may happen during multilateration or patch merging, due to the collinearity of certain node sets.

However, these previous work all assume that the ranging data and anchor positions are correct, and ignore the existence of outlier links and outlier anchors. These outliers

inevitably exist in practice due to the hardware malfunctions, natural interferences, or even malicious attacks.

A school of researchers acknowledge the threat of outliers and try to enhance multilateration with outlier rejection ability, which is called robust multilateration. Ring-overlapping method [Liu et al. 2005] compares distance measurements to rings, finds the region that is most heavily overlapped by rings, and regards the centroid of this region as a trustworthy location estimate. SISR [Kung et al. 2009] is based on weighted multilateration and assigns smaller weights to outlier links than to normal links, because intuitively outlier links have larger deformation and can be identified. LMS [Li et al. 2005] supports robust multilateration based on least median of squares. $C(n, 3)$ methods [Kiyavash and Koushanfar 2007; Wang et al. 2007] find an outlier-free link group by checking each group of three links. Different from the previous researches on robust multilateration, we provide a robust patch merging operation that can reject outliers for both patch merging and multilateration. Based on this robust operation, we further solve the problem of robustly localizing networks which can be sparse or dense.

A recent work [Jian et al. 2010] proposes an outlier link rejection algorithm which can remove outlier links before the execution of a network localization algorithm. This work identifies outlier links based on the enumeration of realizable generic cycles: a generic cycle is outlier-free if it can be realized, and a link is identified as an outlier if it is not contained in any outlier-free generic cycles. In contrast, we reject outlier links based on robust patch merging operation, and we can additionally reject outlier anchors even when multiple outlier anchors collude due to malicious attacks.

8. CONCLUSION

This article focuses on the problem of localizing a wireless sensor network robustly against outlier links and outlier anchors. We proposed a solution RobustLoc which iteratively invokes our robust patch merging operation. Additionally, we addressed the two challenges of (1) isolating non-rejectable outlier links by finding incompatible patch pairs and (2) reliably rejecting multiple outlier anchors which may collude. Our experiments show that our RobustLoc algorithm can retain good localization accuracy with the presence of outliers and meanwhile provide high localization percentages in both sparse networks and dense networks.

REFERENCES

Eren, T., Goldenberg, D. K., Whiteley, W., Yang, Y. R., Morse, A. S., Anderson, B. D. O., and Belhumeur, P. N. 2004. Rigidity, computation, and randomization in network localization. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*.

Foy, W. H. 1976. Position-location solutions by Taylor-series estimation. *IEEE Trans. Aerospace Electron. Syst. 12,* 2, 187–194.

Goldenberg, D. K., Bihler, P., Cao, M., Fang, J., Anderson, B. D. O., Morse, A. S., and Yang, Y. R. 2006. Localization in sparse networks using sweeps. In *Proceedings of the Annual ACM International Conference on Mobile Computing and Networking*.

He, T., Huang, C., Blum, B. M., Stankovic, J. A., and Abdelzaher, T. 2003. Range-free localization schemes for large scale sensor networks. In *Proceedings of the Annual ACM International Conference on Mobile Computing and Networking*.

Horn, B. K. P., Hilden, H., and Negahdaripour, S. 1988. Closed-form solution of absolute orientation using orthonormal matrices. *J. Opti. Soc. Am. A*, 4, 629.

Jian, L.-R., Yang, Z., and Liu, Y.-H. 2010. Beyond triangle inequality: Sifting noisy and outlier distance measurements for localization. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*.

Kannan, A. A., Fidan, B., and Mao, G.-Q. 2011. Robust distributed sensor network localization based on analysis of flip ambiguities. *Wirel. Netw. 17,* 5, 1157–1171.

Kiyavash, N. and Koushanfar, F. 2007. Anti-collusion position estimation in wireless sensor networks. In *Proceedings of the IEEE Conference on Mobile, Ad Hoc and Sensor System*.

Kung, H. T., Lin, C.-K., Lin, T.-H., and Vlah, D. 2009. Localization with snap-inducing shaped residuals (SISR): Coping with errors in measurement. In *Proceedings of the Annual ACM International Conference on Mobile Computing and Networking*.

Li, M. and Liu, Y.-H. 2007. Rendered path: Range-free localization in anisotropic sensor networks with holes. In *Proceedings of the Annual ACM International Conference on Mobile Computing and Networking*. 51–62.

Li, Z., Trappe, W., Zhang, Y., and Nath, B. 2005. Robust statistical methods for securing wireless localization in sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks*. 12.

Lim, H. and C., H. J. 2005. Localization for anisotropic sensor networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*. 138–149.

Liu, D., Ning, P., and Du, W. K. 2005. Attack-resistant location estimation in sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks*.

Mao, G.-Q. and Fidan, B. 2009. *Localization Algorithms and Strategies for Wireless Sensor networks*. IGI Global, Hershey, PA.

Moore, D., Leonard, J., Rus, D., and Teller, S. 2004. Robust distributed network localization with noisy range measurements. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*.

Newsome, J., Shi, E., Song, D., and Perrig, A. 2004. The sybil attack in sensor networks: Analysis & defenses. In *Proceedings of the International Conference on Information Processing in Sensor Networks*.

Niculescu, D. and Nath, B. 2003. DV based positioning in ad hoc networks. *Kluwer J. Telecommun. Syst.*

Priyantha, N. B., Balakrishnan, H., Demaine, E., and Teller, S. 2003. Anchor-free distributed localization in sensor networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*.

Savvides, A., Park, H., and Srivastava, M. B. 2003. The n-hop multilateration primitive for node localization problems. *J. Mobile Netw. Appl. 8*, 4, 443–451.

Shang, Y. and Ruml, W. 2004. Improved MDS-based localization. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*. 2640–2651.

Wang, C., Liu, A., and Ning, P. 2007. Cluster-based minimum mean square estimation for secure and resilient localization in wireless sensor networks. In *Proceedings of the International Conference on Wireless Algorithms Systems and Applications*.

Wang, C. and Xiao, L. 2006. Locating sensors in concave areas. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*. 1–12.

Wang, X.-P., Liu, Y.-H., Yang, Z., Liu, J.-L., and Luo, J. 2010. ETOC: Obtaining robustness in component-based localization. In *Proceedings of the Annual International Conference on Network Protocols*.

Wang, X.-P., Luo, J., Li, S.-S., Dong, D.-Z., and Cheng, W.-F. 2008. Component based localization in sparse wireless ad hoc and sensor networks. In *Proceedings of the Annual International Conference on Network Protocols*.

Whitehouse, K., Karlof, C., Woo, A., Jiang, F., and Culler, D. 2005. The effects of ranging noise on multihop localization: an empirical study. In *Proceedings of the International Conference on Information Processing in Sensor Networks*.

Xiao, B., Chen, H., and Zhou, S. 2008. Distributed localization using a moving beacon in wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst. 19,* 5, 587–600.

Xiao, B., Chen, L., Xiao, Q.-J., and Li, M.-L. 2010a. Reliable anchor-based sensor localization in irregular areas. *IEEE Trans. Mobile Comput. 9*, 60–72.

Xiao, Q.-J., Xiao, B., Cao, J.-N., and Wang, J.-P. 2010b. Multihop range-free localization in anisotropic wireless sensor networks: A pattern-driven scheme. *IEEE Trans. on Mobile Comput. 9*, 1592–1607.

Received April 2011; revised February 2012; accepted February 2012